Progress (100%)

Item 1/20

The fact that tuples belong to sequence types means:

- they are actually lists
- they can be extended using the `.append()` method
- they can be modified using the `del` instruction
- **they can be indexed and sliced like lists**

Next →

Progress (100%)

Item 2/20

What is the output of the following snippet?

```
def fun(x):
    x += 1
    return x

x = 2
x = fun(x + 1)
print(x)
```

- 3
- 5
- 4
- the code is erroneous

← Prev Next →

Progress (100%)

Item 3/20

What is the output of the following snippet?

```
dct = { 'one':'two', 'three':'one', 'two':
v = dct['one']
for k in range(len(dct)):
    v = dct[v]
print(v)
```

- (•) `two`
- ( ) `three`
- ( ) `('one', 'two', 'three')`
- ( ) `one`

← Prev Next →

Progress (100%)

Item 4/20

Assuming that `tuple` is a correctly created tuple, the fact that tuples are immutable means that the following instruction:

```
tuple[1] = tuple[1] + tuple[0]
```

- (●) is illegal
- ( ) can be executed if and only if the tuple contains at least two elements
- ( ) is fully correct
- ( ) may be illegal if the tuple contains strings

← Prev Next →

Progress (100%)

Item 5/20

The following snippet:

```
def func(a, b):
    return a ** a

print(func(2))
```

- will output `4`
- **is erroneous**
- will output `2`
- will return `None`

← Prev Next →

Progress (100%)

Item 6/20

What is the output of the following snippet?

```
def fun(inp=2, out=3):
    return inp * out
print(fun(out=2))
```

- (•) 4
- ( ) 2
- ( ) 6
- ( ) the snippet is erroneous

← Prev Next →

Progress (100%)

Item 7/20

What is the output of the following snippet?

```
def f(x):
    if x == 0:
        return 0
    return x + f(x - 1)

print(f(3))
```

- 3
- 1
- the code is erroneous
- 6

← Prev Next →

Progress (100%)

Item 8/20

What is the output of the following snippet?

```
list =  ['Mary', 'had', 'a', 'little', 'la
def list(lst):
    del lst[3]
    lst[3] = 'ram'
print(list(list))
```

- (●) the snippet is erroneous
- ( ) `['Mary', 'had', 'a', 'little', 'lamb']`
- ( ) `['Mary', 'had', 'a', 'lamb']`
- ( ) `['Mary', 'had', 'a', 'ram']`

← Prev Next →

Progress (100%)

Item 9/20

The following snippet:

```
def func1(a):
    return a ** a
def func2(a):
    return func1(a) * func1(a)
print(func2(2))
```

- is erroneous
- will output `4`
- will output `2`
- **will output `16`**

← Prev Next →

Progress (100%)

Item 10/20

What is the output of the following snippet?

```
def fun(x):
    if x % 2 == 0:
        return 1
    else:
        return

print(fun(fun(2)) + 1)
```

- None
- 2
- 1
- **the code will cause a runtime error**

← Prev Next →

Item 11/20

A function defined in the following way:

```
def function(x=0):
    return x
```

- must be invoked without arguments
- may be invoked with any number of arguments (including zero)
- **may be invoked without any argument, or with just one**
- must be invoked with exactly one argument

← Prev Next →

Progress (100%)

Item 12/20

Which of the following lines properly starts a function using two parameters, both with zeroed default values?

- `fun fun(a, b=0):`
- `def fun(a=b=0):`
- `fun fun(a=0, b):`
- **(selected)** `def fun(a=0, b=0):`

← Prev Next →

Item 13/20

What code would you insert into the commented line to obtain the output that reads:

Expected output:

```
a
b
c
```

Code:

```
dct = { }
lst = ['a', 'b', 'c', 'd']
for i in range(len(lst) - 1):
    dct[lst[i]] = ( lst[i], )
for i in sorted(dct.keys()):
    k = dct[i]
    # insert your code
```

- `print(k["0"])`
- `print(k)`
- `print(k['0'])`
- (●) `print(k[0])`

← Prev Next →

Progress (100%)

Item 14/20

A built-in function is a function which:

- has been placed within your code by another programmer
- has to be imported before use
- comes with Python, and is an integral part of Python
- is hidden from programmers

← Prev Next →

Progress (100%)

Item 15/20

What is the output of the following snippet?

```
tup = (1, 2, 4, 8)
tup = tup[1:-1]
tup = tup[0]
print(tup)
```

- the snippet is erroneous
- (2)
- 2
- (2,)

← Prev Next →

Progress (100%)

Item 16/20

What is the output of the following snippet?

```
def any():
    print(var + 1, end='')
var = 1
any()
print(var)
```

- 11
- 12
- 21
- 22

← Prev Next →

Progress (100%)

Item 17/20

What is the output of the following snippet?

```
def fun(x):
    global y
    y = x * x
    return y

fun(2)
print(y)
```

- (•) 4
- ( ) None
- ( ) the code will cause a runtime error
- ( ) 2

← Prev Next →

Item 18/20

Which of the following statements is false?

- The `None` value can be assigned to variables
- The `None` value can be compared with variables
- The `None` value cannot be used as an argument of arithmetic operators
- (●) The `None` value may not be used outside functions

← Prev Next →

Progress (100%)

Item 19/20

What is the output of the following snippet?

```
def fun(x, y, z):
    return x + 2 * y + 3 * z

print(fun(0, z=1, y=3))
```

- the snippet is erroneous
- 3
- 9
- 0

← Prev Next →

Progress (100%)

Item 20/20

Which of the following lines properly starts a parameterless function definition?

- `def fun():`
- `fun function():`
- `function fun():`
- `def fun:`

Prev